Искусственные Нейронные сети. Персептрон

Гончаров Павел kaliostrogoblin3@gmail.com
Нестереня Игорь nesterione@gmail.com

Как заставить машину мыслить?

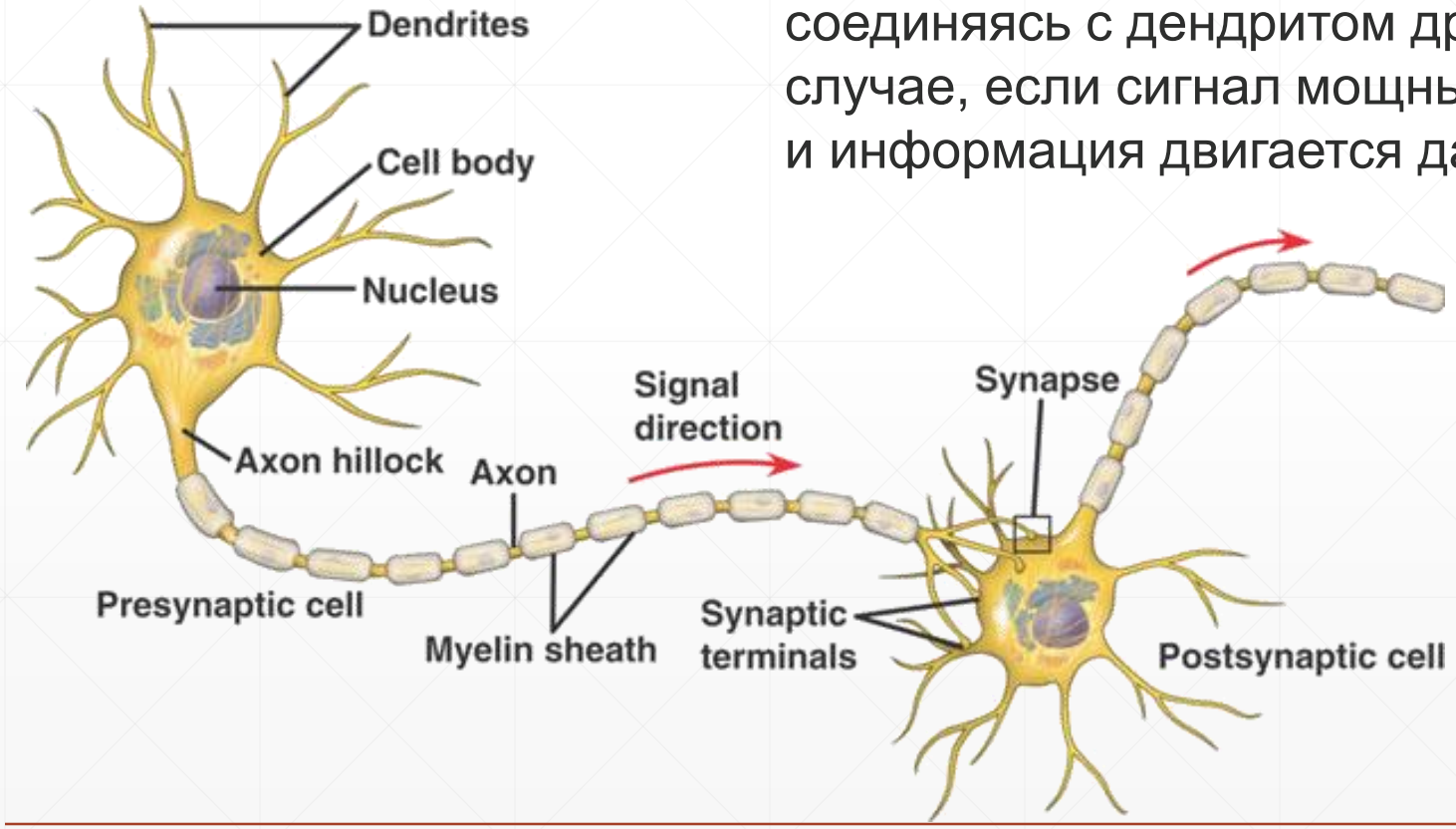
Ученые уже в 50-х годах прошлого века задумывались о том, как заставить ЭВМ мыслить. Одним из решений было **создать упрощенную копию нейрона человека**.

Человеческий мозг пластичен и обладает рядом **полезных свойств**:

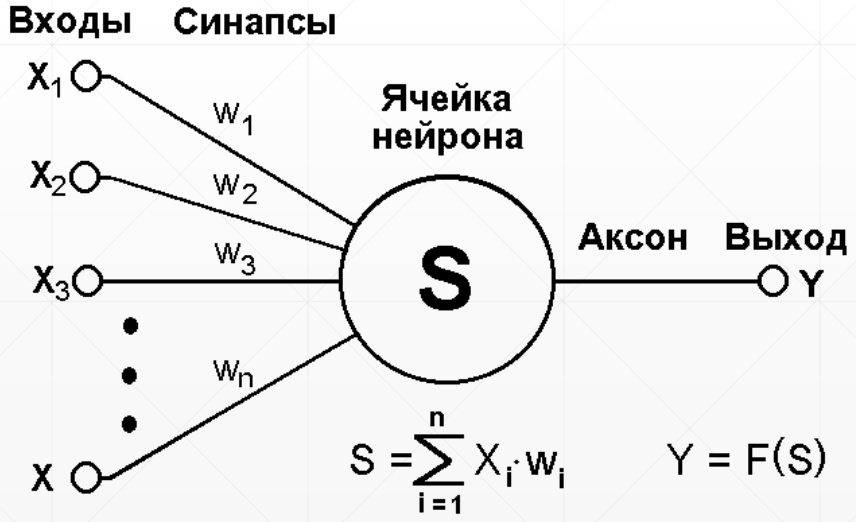
- способность выполнять в параллельном режиме логические, распознающие и оптимизирующие функции;
- давать приемлемый ответ даже при неполных или ошибочных входных данных;
- устойчиво работать при сбоях или выходе из строя одного или даже группы нейронов (Луи Пастер);
- обучаться, т. е. улучшать свои характеристики в процессе обработки данных.

Модель биологического нейрона

Биологический нейрон – это клетка, состоящая из тела нейрона, **сомы**, сигналы извне поступают к телу через отростки – **дендриты**. Информация от нейрона к нейрону проходит с помощью **аксона**. Аксон одного нейрона, соединяясь с дендритом другого, создает связь – синапс, в случае, если сигнал мощный, происходит активация – **спайк** и информация двигается дальше.



Модель искусственного нейрона

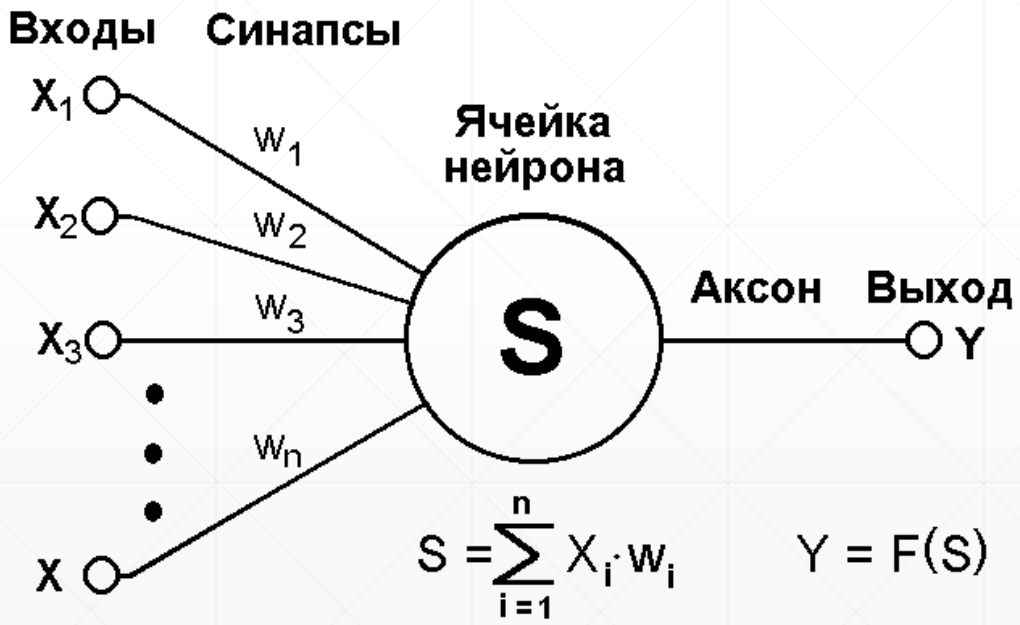


Немного истории

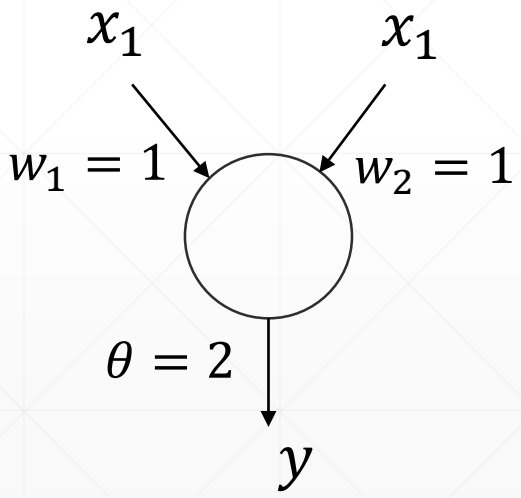
1. В 1907 году Santiago Ramón y Cajal и Camillo Golgi получили Нобелевскую премию за теорию нейронных систем.
2. Позже, в 1943 году, ученые MacKallok и Pitts предложили математическую модель биологического нейрона.
3. В 1949 году D.Hebb сформулировал правило обучения нейронов : при повторном или настойчивом возбуждении одним нейроном другого их синаптическая связь усиливается.
4. Эти идеи развил в своих работах Фрэнк Розенблатт (1958). И 23 июня 1960 года в Корнеллском университете, был продемонстрирован первый нейрокомпьютер – «Марк-1», который был способен распознавать некоторые буквы английского алфавита.
5. в 1969 г. Марвин Минский и Сеймур Паперт опубликовали книгу «Перцептроны», в которой разгромили персептрон. Так как Розенблатт трагически погиб во время плавания на яхте в день своего рождения, то не смог ответить на критику и финансирование исследований было урезано.

Модель искусственного нейрона

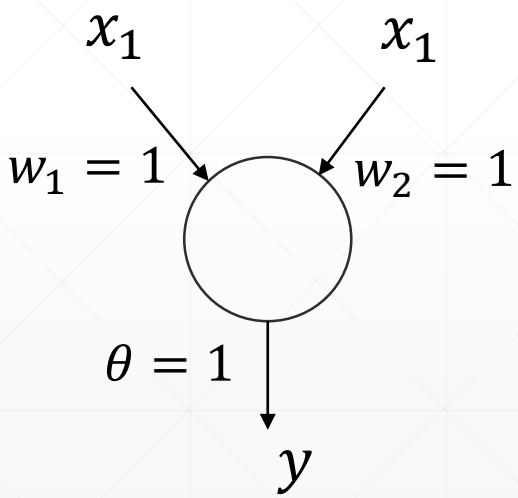
Искусственный нейрон – это упрощенный вариант биологического нейрона. Нейрон представляет собой взвешенный сумматор: мы умножаем входной набор признаков X на вектор параметров W, после чего суммируем результат – S. Полученное значение проходит через **пороговую функцию активации F**.



Нейрон как логическая функция



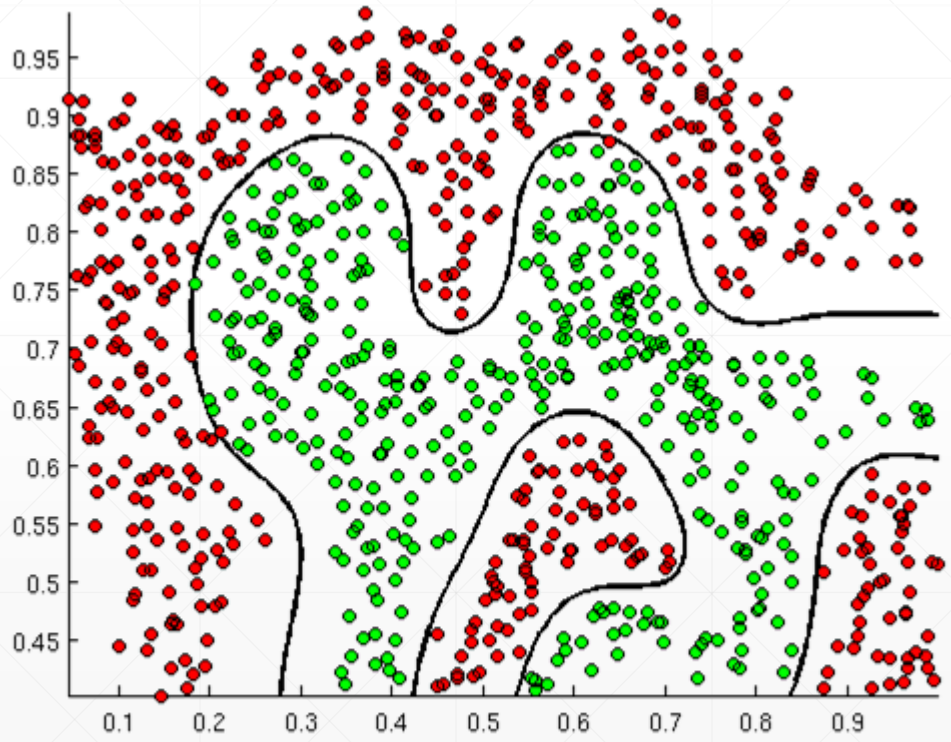
«AND»



«OR»

Предпосылки: нелинейность

Представим себе задачу классификации, где решением является **сложная нелинейная функция**. Для нее нужно найти гипотезу с полиномом высокой степени.

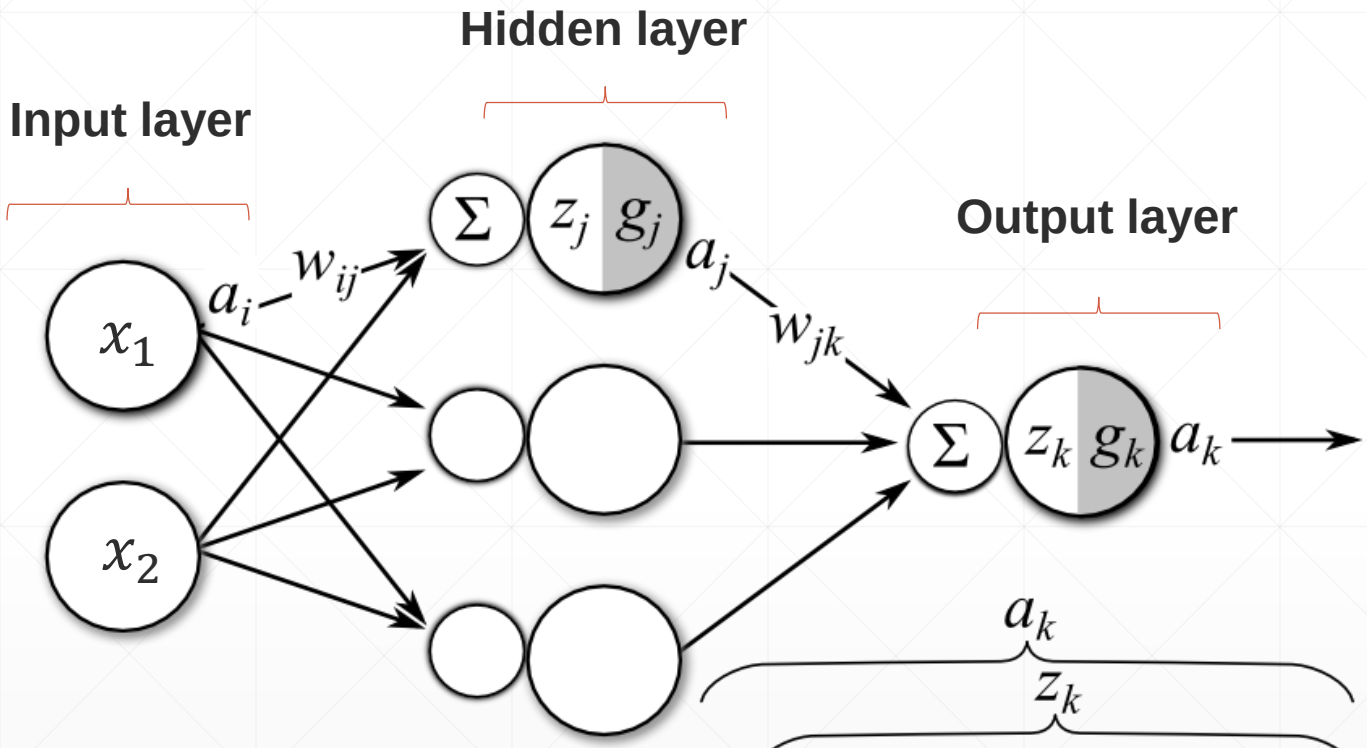


Теперь представим, что входной набор признаков – **изображения** 50x50 = 2500 признаков.

Если взять полином второй степени (квадраты исходных признаков), то количество входных значений ≈ 3 млн.

Выход – искусственные нейронные сети (ИНС), многослойный персептрон.

Архитектура многослойной ИНС



g – функция активации, сигмоид

$$h_w(x) = a_k = \frac{1}{1 + e^{-z_k}}$$

Feed forward propagation →

$$a_k = g_k(b_k + \Sigma_j g_j(b_j + \Sigma_i a_i w_{ij}) w_{jk})$$

a_k = $g_k(\ldots)$, z_k = $b_k + \Sigma_j g_j(\ldots)$, a_j = $g_j(\ldots)$, z_j = $b_j + \Sigma_i a_i w_{ij}$

Нужно для градиентного спуска:

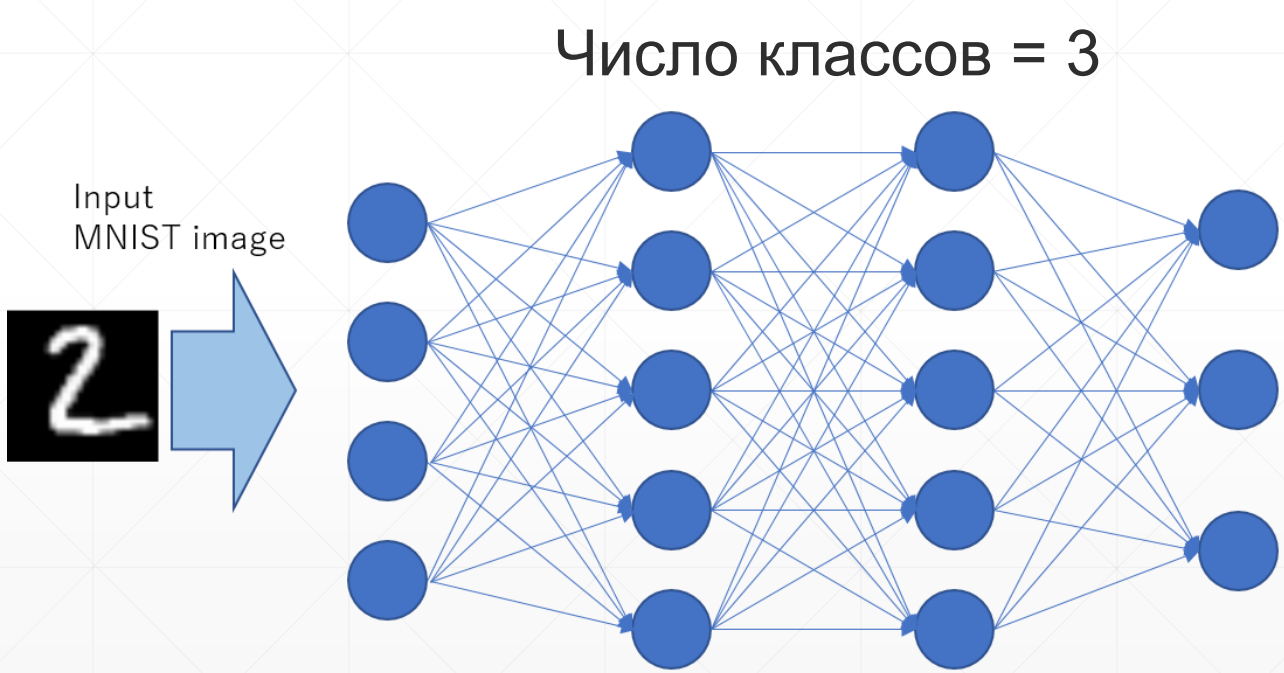
a – хранит все активации нейронов

z – состояния ячеек нейронов

Целевая функция, cost function

Функция ошибки для логистической регрессии:

$$Loss = -ylog\big(h_w(x)\big) - (1-y)\log(1-h_w(x))$$



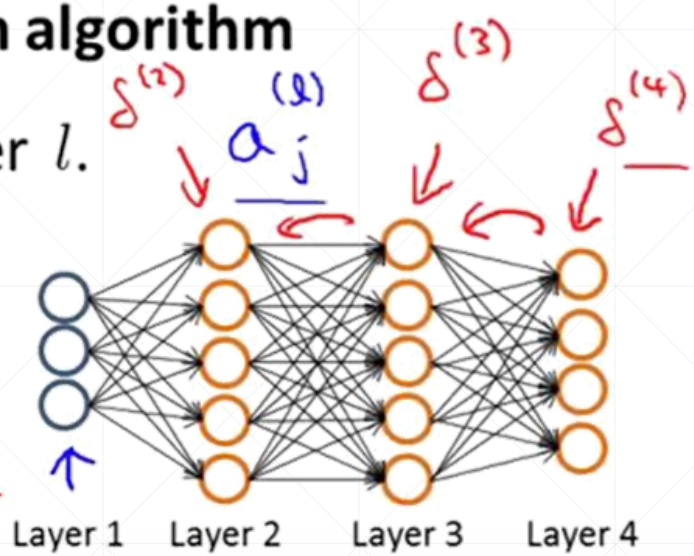
Задача – найти минимум cost: $\boldsymbol{min\ Cost}$

Для ИНС многоклассовой классификации целевая функция – это средняя величина $Loss$ на обучающем примере, где $Loss$ – это сумма расхождений между logits и labels.

Обратное распространение ошибки

Gradient computation: Backpropagation algorithm

Intuition: $\delta_j^{(l)}$ = "error" of node j in layer l.



For each output unit (layer L = 4)

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

$(h_\Theta(x))_j$ $\delta^{(4)} = a^{(4)} - y$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$$

$a^{(3)} .* (1 - a^{(3)})$

$a^{(2)} .* (1 - a^{(2)})$

(No $\delta^{(1)}$)

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

(ignore λ; if $\lambda = 0$)

Ссылка: https://www.coursera.org/learn/machine-learning/lecture/1z9WW/backpropagation-algorithm

Алгоритм обратного распространения ошибки

Алгоритм:

- задать начальное значение параметров W, близкие к нулю значения (**но не нули!**)
- Задать число эпох $epochs$
- задать ***скорость обучения*** α
- $For\ i\ in\ range(epochs)$:
  - Задать начальные смещения для параметров $delta\_W = 0, delta\_b = 0$. Смещения задаются для каждого слоя (несколько матриц параметров)
  - Для каждого примера (x,y) в обучающей выборке:
    - $(a, z) = forward\_prop(x)$ [слайд 7] - все активации и состояния
    - $\delta = backprop(y, a, z)$ [слайд 9] - ошибки на каждом слое
    - $delta\_W = delta\_W + \delta\_W$
    - $delta\_b = delta\_b + \delta\_b$
  - $W = W - \frac{\alpha}{m} delta\_W$
  - $b = b - \frac{\alpha}{m} delta\_b$

Пора распространять ошибки обратно!